```c
#include<stdio.h>
#include<stdlib.h>

#define eps 5e-5

int mem = 0, memMax = 0;
FILE *fm;

struct QDD
{
    struct no *n;
    struct lista *l;
};

struct no
{
    struct lista *l;
    unsigned short tipo, nivel;  //Tipos 0 - número    1 - V    2 - R    3 - C
    float re, im;
    struct no *el, *th;
};

struct lista
{
    struct lista *l;
    struct no *n;
};



typedef struct QDD QDD;

typedef struct no no;

typedef struct lista lista;

typedef unsigned short Short;



void aumenta_memoria(int m)
{
    mem += m;
    fprintf(fm,"\nMemUP: %d\t\t\t %d",mem,m);
    if(memMax<mem)
    {
        memMax = mem;
        fprintf(fm,"\t\tMemMax");
    }
}

void diminui_memoria(int m)
{
    mem -= m;
    fprintf(fm,"\n\tMemDOWN: %d\t\t%d",mem,-m);
}



QDD* cria_QDD()
{
    QDD *Q;
    Q = malloc(sizeof(QDD));
    aumenta_memoria(sizeof(QDD));
    Q->n = NULL;
    Q->l = NULL;
```

```
67      return Q;
68  }
69
70  no* cria_no(Short tipo, Short nivel, float re, float im)
71  {
72      no* n;
73      n = malloc(sizeof(no));
74      aumenta_memoria(sizeof(no));
75      n->l = NULL;
76      n->tipo = tipo;
77      n->nivel = nivel;
78      n->re = re;
79      n->im = im;
80      n->el = NULL;
81      n->th = NULL;
82      return n;
83  }
84
85  no* cria_no_vazio()
86  {
87      no* n;
88      n = cria_no(0,0,0,0);
89      return n;
90  }
91
92  lista* cria_no_lista()
93  {
94      lista *l;
95      l = malloc(sizeof(lista));
96      aumenta_memoria(sizeof(lista));
97      if(l == NULL)
98          exit(0);
99      l->l = NULL;
100     l->n = NULL;
101     return l;
102 }
103
104
105
106 void libera_no_QDD(QDD *Q)
107 {
108     diminui_memoria(sizeof(QDD));
109     free(Q);
110 }
111
112 void libera_no(no *n)
113 {
114     diminui_memoria(sizeof(no));
115     free(n);
116 }
117
118 void libera_no_lista(lista *l)
119 {
120     diminui_memoria(sizeof(lista));
121     free(l);
122 }
123
124 void libera_lista(lista *l)
125 {
126     lista *lc;
127     while(l != NULL)
128     {
129         lc = l->l;
130         libera_no_lista(l);
131         l = lc;
132     }
```

```
}



void mostra_lista(lista *l)
{
    lista *lc;
    Short ligacao = 0;
    lc = l;
    for(lc = l; lc != NULL; lc = lc->l)
    {
        printf("\tLigacao %u: %d\n",ligacao,lc->n);
        ligacao++;
    }
}

void fmostra_lista(FILE *fp, lista *l)
{
    lista *lc;
    Short ligacao = 0;
    lc = l;
    for(lc = l; lc != NULL; lc = lc->l)
    {
        fprintf(fp,"\tLigacao %u: %d\n",ligacao,lc->n);
        ligacao++;
    }
}

void mostra_no(no *n)
{
    printf("\nEndereco: %d\n",n);
    if(n->l != NULL)
    {
        printf("Ligacoes anteriores:\n");
        mostra_lista(n->l);
    }
    printf("Tipo");
    switch(n->tipo)
    {
        case 0:
        printf(": Numero\n");
        printf("%f %f \n",n->re,n->im);
        break;

        case 1:
        printf("/nivel: V%d\n",n->nivel);
        break;

        case 2:
        printf("/nivel: R%d\n",n->nivel);
        break;

        case 3:
        printf("/nivel: C%d\n",n->nivel);
        break;
    }
    if((n->el != NULL)||(n->th != NULL))
    {
        printf("Ligacoes posteriores\n");
        printf("\telse: %d\n",n->el);
        printf("\tThen: %d\n",n->th);
    }
    printf("\n");
}

void fmostra_no(FILE *fp, no *n)
```

```
{
    fprintf(fp,"\nEndereco: %d\n",n);
    if(n->l != NULL)
    {
        fprintf(fp,"Ligacoes anteriores:\n");
        fmostra_lista(fp,n->l);
    }
    fprintf(fp,"Tipo");
    switch(n->tipo)
    {
        case 0:
        fprintf(fp,": Numero\n");
        fprintf(fp,"%f %f \n",n->re,n->im);
        break;

        case 1:
        fprintf(fp,"/nivel: V%d\n",n->nivel);
        break;

        case 2:
        fprintf(fp,"/nivel: R%d\n",n->nivel);
        break;

        case 3:
        fprintf(fp,"/nivel: C%d\n",n->nivel);
        break;
    }
    if((n->el != NULL)||(n->th != NULL))
    {
        fprintf(fp,"Ligacoes posteriores\n");
        fprintf(fp,"\telse: %d\n",n->el);
        fprintf(fp,"\tThen: %d\n",n->th);
    }
}

void mostra_lista_com_no(lista *l)
{
    lista *lc;
    Short ligacao = 0;
    lc = l;
    printf("\n");
    for(lc = l; lc != NULL; lc = lc->l)
    {
        printf("\tLigacao %u: %d\n",ligacao,lc->n);
        mostra_no(lc->n);
        ligacao++;
    }
}

void fmostra_lista_com_no(FILE *fp, lista *l)
{
    lista *lc;
    Short ligacao = 0;
    lc = l;
    fprintf(fp,"\n");
    for(lc = l; lc != NULL; lc = lc->l)
    {
        fprintf(fp,"\n\tLigacao %u: %d\n",ligacao,lc->n);
        fmostra_no(fp,lc->n);
        ligacao++;
    }
}

void mostra_arvore_ineficiente(no *n)
{
    if(n == NULL)
```

```
265         return;
266     mostra_no(n);
267     mostra_arvore_ineficiente(n->el);
268     mostra_arvore_ineficiente(n->th);
269 }
270 
271 lista* enlista_QDD(QDD *Q)
272 {
273     lista *l, *la, *lc, *lf;
274     no *n;
275 
276     l = cria_no_lista();
277     la = cria_no_lista();
278     la->n = Q->n;
279     lf = l;
280 
281     while(la != NULL)
282     {
283         n = la->n;
284         for(lc = l; lc != NULL; lc = lc->l)
285             if(lc->n == n)
286                 break;
287         if(lc == NULL)
288         {
289             lf->l = cria_no_lista();
290             lf = lf->l;
291             lf->n = n;
292 
293             la->n = n->th;
294             lc = cria_no_lista();
295             lc->n = n->el;
296             lc->l = la;
297             la = lc;
298         }
299         else
300         {
301             lc = la->l;
302             libera_no_lista(la);
303             la = lc;
304         }
305     }
306     l->n = Q->n->l->n;
307     return l;
308 }
309 
310 void mostra_QDD(QDD *Q)
311 {
312     lista *l;
313     l = enlista_QDD(Q);
314     mostra_lista_com_no(l);
315     printf("\n");
316     mostra_lista(Q->l);
317     libera_lista(l);
318 }
319 
320 void fmostra_QDD(FILE *fp,QDD *Q)
321 {
322 
323     lista *l;
324     l = enlista_QDD(Q);
325     fmostra_lista_com_no(fp,l);
326     fprintf(fp,"\n");
327     fmostra_lista(fp,Q->l);
328     libera_lista(l);
329 }
330 
```

```
331 void fmostra_QDD_sozinho(QDD *Q, char *nome)
332 {
333     FILE *fp;
334     fp = fopen(nome,"w");
335     fmostra_QDD(fp,Q);
336     fclose(fp);
337 }
338
339
340
341 void conecta_UM(no *n1, no *n2, Short lado)
342 {
343     lista *l;
344
345     if(lado)
346         n1->th = n2;
347     else
348         n1->el = n2;
349
350     l = cria_no_lista();
351     l->n = n1;
352     l->l = n2->l;
353     n2->l = l;
354 }
355
356 void conecta_DOIS(no *n, no *el, no *th)
357 {
358     conecta_UM(n,el,0);
359     conecta_UM(n,th,1);
360 }
361
362 Short desconecta_UM(no *n1, no *n2)
363 {
364     lista *l, *lc, *laux;
365     Short lado;
366     if(n1->el == n2)
367     {
368         n1->el = NULL;
369         lado = 0;
370     }
371     else
372     {
373         n1->th = NULL;
374         lado = 1;
375     }
376
377     l = cria_no_lista();
378     l->l = n2->l;
379     for(lc = l; lc->l != NULL; lc = lc->l)
380     {
381         if(lc->l->n == n1)
382         {
383             laux = lc->l;
384             lc->l = laux->l;
385             libera_no_lista(laux);
386             break;
387         }
388     }
389     n2->l = l->l;
390     libera_no_lista(l);
391     return lado;
392 }
393
394 void desconecta_DOIS(no *n)
395 {
396     desconecta_UM(n,n->el);
```

```
397     desconecta_UM(n,n->th);
398 }
399
400 void transfere_conexao(no *n1, no *n2)
401 {
402     no *n;
403     Short lado;
404     while(n2->l != NULL)
405     {
406         n = n2->l->n;
407         lado = desconecta_UM(n,n2);
408         conecta_UM(n,n1,lado);
409     }
410 }
411
412
413
414 void libera_QDD(QDD *Q)
415 {
416     lista *l, *lc;
417     l = enlista_QDD(Q);
418     no *n1, *n2;
419     for(lc = l; lc != NULL; lc = lc->l)
420     {
421         n1 = lc->n;
422         while(n1->l != NULL)
423         {
424             n2 = n1->l->n;
425             desconecta_UM(n2,n1);
426         }
427     }
428
429     for(lc = l; lc != NULL; lc = lc->l)
430         libera_no(lc->n);
431
432     libera_lista(l);
433     libera_lista(Q->l);
434
435     libera_no_QDD(Q);
436 }
437
438
439
440 void completa_QDD(no *n, Short r, Short c, Short exp, Short **M, lista **L)
441 {
442     Short ind1, ind2;
443     if((n->tipo == 3)&&(exp == 1))
444     {
445         ind1 = M[r][c];
446         ind2 = M[r][c+1];
447         conecta_DOIS(n,L[ind1]->n,L[ind2]->n);
448     }
449     else
450     {
451         if(n->tipo == 2)
452         {
453             completa_QDD(n->el,r,c,exp,M,L);
454             completa_QDD(n->th,r+exp,c,exp,M,L);
455         }
456         if(n->tipo == 3)
457         {
458             completa_QDD(n->el,r,c,exp/2,M,L);
459             completa_QDD(n->th,r,c+exp,exp/2,M,L);
460         }
461     }
462 }
```

```
463
464 QDD* le_matriz(char *nome)
465 {
466     Short i, j, k;
467
468     FILE *fp;
469     fp = fopen(nome,"r");
470
471     Short N1, N2, N3;
472     fscanf(fp,"%hu %hu\n%hu\n",&N1, &N2, &N3);
473
474
475     lista **L;
476     float re, im;
477     L = malloc(N3*sizeof(lista*));
478     aumenta_memoria(N3*sizeof(lista*));
479     for(i=0; i<N3; i++)
480     {
481         L[i] = cria_no_lista();
482         fscanf(fp,"%f %f",&re, &im);
483         L[i]->n = cria_no(0,N1,re,im);
484     }
485     fscanf(fp,"\n");
486     for(i=0; i<N3-1; i++)
487         L[i]->l = L[i+1];
488
489     Short **M;
490     M = malloc(N2*sizeof(Short*));
491     aumenta_memoria(N2*sizeof(Short*));
492     for(i=0; i<N2; i++)
493     {
494         M[i] = malloc(N2*sizeof(Short));
495         aumenta_memoria(N2*sizeof(Short));
496         for(j=0; j<N2; j++)
497             fscanf(fp,"%hu",&M[i][j]);
498     }
499
500     no **N;
501     N = malloc((N2*N2-1)*sizeof(no*));
502     aumenta_memoria((N2*N2-1)*sizeof(no*));
503
504     Short exp, ind;
505     exp = 1;
506     ind = 0;
507     for(i=0; i<N1; i++)
508     {
509         for(j=2; j<=3; j++)
510         {
511             for(k=0; k<exp; k++)
512             {
513                 N[ind] = cria_no(j,i,0,0);
514                 ind++;
515             }
516             exp *= 2;
517         }
518     }
519
520     for(i=0; i<(N2*N2-1)/2; i++)
521         conecta_DOIS(N[i],N[2*i+1],N[2*i+2]);
522
523     completa_QDD(N[0],0,0,N2/2,M,L);
524
525     QDD *Q;
526     Q = cria_QDD();
527     Q->n = cria_no_vazio();
528     conecta_UM(Q->n,N[0],1);
```

```
529      Q->n = N[0];
530      Q->l = L[0];
531
532      free(N);
533      diminui_memoria((N2*N2-1)*sizeof(no*));
534      free(L);
535      diminui_memoria(N3*sizeof(lista*));
536      for(i=0; i<N2; i++)
537      {
538          free(M[i]);
539          diminui_memoria(N2*sizeof(Short));
540      }
541      free(M);
542      diminui_memoria(N2*sizeof(Short*));
543      free(fp);
544
545      return Q;
546  }
547
548  lista* copia_lista(lista *l1)
549  {
550      lista *l2, *lc, *lc2;
551      l2 = cria_no_lista();
552      lc2 = l2;
553      for(lc = l1; lc != NULL; lc = lc->l)
554      {
555          lc2->l = cria_no_lista();
556          lc2 = lc2->l;
557          lc2->n = lc->n;
558      }
559      return l2;
560  }
561
562  void reduz_QDD(QDD *Q)
563  {
564      no *n1, *n2, *nc;
565      lista *l, *laux, *lnc1, *lnc2, *lr, *lrc, *lf;
566      Short mudou, saida, del, regra;
567      l = copia_lista(Q->l);
568      while(l->l != NULL)
569      {
570          nc = l->l->n;
571          del = 1;
572          do
573          {
574              mudou = 0;
575              for(regra = 2; regra >= 1; regra--)
576              {
577                  do
578                  {
579                      saida = 0;
580                      for(lnc1 = nc->l; lnc1->l != NULL; lnc1 = lnc1->l)
581                      {
582                          n1 = lnc1->n;
583                          for(lnc2 = lnc1->l; lnc2 != NULL; lnc2 = lnc2->l)
584                          {
585                              n2 = lnc2->n;
586                              switch(regra)
587                              {
588                                  case 1:
589                                  if(n1 == n2)
590                                  {
591                                      desconecta_DOIS(n2);
592                                      transfere_conexao(nc,n2);
593                                      libera_no(n2);
594
```

```
595                             mudou = 1;
596                             del = 0;
597                             saida = 1;
598                         }
599                         break;
600 
601                         case 2:
602                         if(n1        != n2       )
603                         if(n1->tipo  == n2->tipo )
604                         if(n1->nivel == n2->nivel)
605                         if(n1->el    == n2->el   )
606                         if(n1->th    == n2->th   )
607                         {
608                             lr = cria_no_lista();
609                             lr->n = n1;
610                             lrc = cria_no_lista();
611                             lrc->n = n2;
612                             lr->l = lrc;
613 
614                             while(lnc2->l != NULL)
615                             {
616                                 lnc2 = lnc2->l;
617                                 n2 = lnc2->n;
618 
619                                 if(n1        != n2       )
620                                 if(n1->tipo  == n2->tipo )
621                                 if(n1->nivel == n2->nivel)
622                                 if(n1->el    == n2->el   )
623                                 if(n1->th    == n2->th   )
624                                 {
625                                     if(n1->el == n1->th)
626                                     {
627                                         for(lrc = lr; lrc->l != NULL; lrc = lrc->l)
628                                             if(lrc->l->n == n2)
629                                                 break;
630                                         if(lrc->l != NULL)
631                                             continue;
632                                     }
633                                     lrc->l = cria_no_lista();
634                                     lrc = lrc->l;
635                                     lrc->n = n2;
636                                 }
637                             }
638 
639                             while(lr->l != NULL)
640                             {
641                                 lrc = lr->l;
642                                 n2 = lrc->n;
643 
644                                 desconecta_DOIS(n2);
645                                 transfere_conexao(n1,n2);
646                                 libera_no(n2);
647 
648                                 lr->l = lrc->l;
649                                 libera_no_lista(lrc);
650                             }
651                             libera_no_lista(lrc);
652 
653                             if(n1->el != n1->th)
654                             {
655                                 for(lf = l; lf->l != NULL; lf = lf->l);
656                                 lf->l = cria_no_lista();
657                                 lf = lf->l;
658                                 lf->n = n1;
659                             }
660 
```

```
                                    mudou = 1;
                                    saida = 1;
                                }
                                break;
                            }
                            if(saida)
                                break;
                        }
                        if(saida)
                            break;
                    }
                }
                while(saida);
            }
        }
        while(mudou);

        if(del)
        {
            laux = l->l;
            libera_no_lista(l);
            l = laux;
        }
        else
        {
            laux = l->l;
            l->l = laux->l;
            laux->l = NULL;

            for(lf = l; lf->l != NULL; lf = lf->l);
            lf->l = laux;
        }
    }
    libera_no_lista(l);
}

no* copia_no(no *n1)
{
    no *n2;
    n2 = cria_no(n1->tipo,n1->nivel,n1->re,n1->im);
    return n2;
}

lista* copia_lista_sem_cabeca(lista *l1)
{
    lista *l2, *laux;
    l2 = copia_lista(l1);
    laux = l2;
    l2 = laux->l;
    libera_no_lista(laux);
    return l2;
}

QDD* copia_QDD(QDD *Q1)
{
    lista *l1, *l2, *lc1a, *lc2a, *lc1b, *lc2b;
    no *n1, *n2, *nf, *nt1, *nt2;

    l1 = enlista_QDD(Q1);
    l2 = copia_lista_sem_cabeca(l1);
    for(lc2a = l2; lc2a != NULL; lc2a = lc2a->l)
        lc2a->n = copia_no(lc2a->n);

    lc1a = l1;
    lc2a = l2;
    do
```

```
727     {
728         n1 = lc1a->n;
729         n2 = lc2a->n;
730 
731         if(n1->el != NULL)
732         {
733             nf = n1->el;
734 
735             lc1b = l1;
736             lc2b = l2;
737             do
738             {
739                 nt1 = lc1b->n;
740                 nt2 = lc2b->n;
741 
742                 lc1b = lc1b->l;
743                 lc2b = lc2b->l;
744             }
745             while(nt1 != nf);
746 
747             conecta_UM(n2,nt2,0);
748         }
749 
750         if(n1->th != NULL)
751         {
752             nf = n1->th;
753 
754             lc1b = l1;
755             lc2b = l2;
756             do
757             {
758                 nt1 = lc1b->n;
759                 nt2 = lc2b->n;
760 
761                 lc1b = lc1b->l;
762                 lc2b = lc2b->l;
763             }
764             while(nt1 != nf);
765 
766             conecta_UM(n2,nt2,1);
767         }
768 
769         lc1a = lc1a->l;
770         lc2a = lc2a->l;
771     }
772     while(lc1a != NULL);
773 
774     QDD *Q2;
775     Q2 = cria_QDD();
776     Q2->n = l2->l->n;
777 
778     lc2a = l2;
779     while(lc2a->l != NULL)
780     {
781         lc2b = lc2a->l;
782         if(lc2b->n->tipo == 0)
783             lc2a = lc2b;
784         else
785         {
786             lc2a->l = lc2b->l;
787             libera_no_lista(lc2b);
788         }
789     }
790 
791     Q2->l = l2->l;
792     libera_no_lista(l2);
```

```
793     libera_lista(l1);
794
795     return Q2;
796 }
797
798 no* produto_complexo(no *n1, no *n2)
799 {
800     no *n;
801     float re, im;
802     re = (n1->re)*(n2->re)-(n1->im)*(n2->im);
803     im = (n1->re)*(n2->im)+(n1->im)*(n2->re);
804     n = cria_no(0,n2->nivel,re,im);
805     return n;
806 }
807
808 Short compara(no *n1, no *n2)
809 {
810     float re, im;
811     re = (n1->re)-(n2->re);
812     im = (n1->im)-(n2->im);
813     if(re<-eps)
814         return 0;
815     if(re>eps)
816         return 0;
817     if(im<-eps)
818         return 0;
819     if(im>eps)
820         return 0;
821     return 1;
822 }
823
824 void reduz_lista(lista *l)
825 {
826     no *n1, *n2;
827     lista *lc1, *lc2, *laux;
828     lc1 = l;
829     for(lc1 = l; lc1->l != NULL; lc1 = lc1->l)
830     {
831         n1 = lc1->n;
832         lc2 = lc1;
833         while(lc2->l != NULL)
834         {
835             n2 = lc2->l->n;
836             if(compara(n1,n2))
837             {
838                 transfere_conexao(n1,n2);
839                 laux = lc2->l;
840                 lc2->l = laux->l;
841                 libera_no_lista(laux);
842                 libera_no(n2);
843             }
844             else
845                 lc2 = lc2->l;
846         }
847         if(lc1->l == NULL)
848             break;
849     }
850 }
851
852 QDD* produto_tensorial(QDD *Q1, QDD *Q2)
853 {
854     QDD *Q;
855     Short nqbit;
856     lista *l;
857
858     Q = copia_QDD(Q1);
```

```
    reduz_QDD(Q);
    nqbit = Q->l->n->nivel;
    l = Q->l;
    Q->l = NULL;

    QDD *Q2a;
    lista *l2, *lc;
    Q2a = copia_QDD(Q2);
    reduz_QDD(Q2a);
    l2 = enlista_QDD(Q2a);
    for(lc = l2; lc != NULL; lc = lc->l)
        (lc->n->nivel) += nqbit;
    libera_lista(l2);

    QDD *Q2b;
    no *n1, *n2, *naux, *n0;
    n0 = cria_no_vazio();
    while(l != NULL)
    {
        n1 = l->n;
        if(compara(n0,n1))
        {
            l2 = l->l;
            l->l = Q->l;
            Q->l = l;
            l = l2;
        }
        else
        {
            Q2b = copia_QDD(Q2a);
            for(lc = Q2b->l; lc != NULL; lc = lc->l)
            {
                n2 = lc->n;
                naux = produto_complexo(n1,n2);
                transfere_conexao(naux,n2);
                lc->n = naux;
                libera_no(n2);
            }

            n2 = Q2b->n;
            naux = Q2b->n->l->n;
            desconecta_UM(naux,n2);
            libera_no(naux);

            transfere_conexao(n2,n1);
            libera_no(n1);

            lc = Q2b->l;
            while(lc->l != NULL)
                lc = lc->l;
            lc->l = Q->l;
            Q->l = Q2b->l;
            libera_no_QDD(Q2b);
        }
        lc = l->l;
        libera_no_lista(l);
        l = lc;
    }

    libera_QDD(Q2a);

    reduz_lista(Q->l);
    reduz_QDD(Q);

    return Q;
}
```

```
925
926
927
928  int main()
929  {
930      fm = fopen("MemReport.txt","w");
931
932      QDD *Q1, *Q2, *Q3, *Q4;
933      Q1 = le_matriz("Had8.txt");
934      reduz_QDD(Q1);
935
936      Q2 = copia_QDD(Q1);
937
938      Q3 = produto_tensorial(Q1,Q2);
939
940      libera_QDD(Q1);
941      libera_QDD(Q2);
942      libera_QDD(Q3);
943
944      fprintf(fm,"\n\nMemMax: %d",memMax);
945      free(fm);
946  }
```